Лабораторная работа №2

Тема: Чтение и запись графических изображений в формате **BMP**

Задания:

1. Написать функцию для отображения (чтения с диска) в заданной области окна изображения в формате BMP.

   int ShowBitMap(HWND hWnd, RECT& r, HBITMAP hBit, int x, int y)
   //Функция отображает рисунок в заданной позиции окна
   //hWnd - дискриптор окна, куда выводится изображение
   //r – область в окне, куда выводится изображение
   //hBit - дискриптор рисунка
   //(x,y) - координаты левого верхнего угла изображения в окне вывода

2. Написать функцию для записи на диск изображения заданной области окна в формате BMP.

   int ClientToBmp(HWND hWnd, RECT& r, char *Name)
   //Сохранение рабочей области окна в файле Name.bmp
   //hWnd - дискриптор окна, рабочая область которого сохраняется
   //r – область в окне, которая сохраняется в файле
   //Name - имя файла для сохранения

3. Тестирование функций **ShowBitMap** и **ClientToBmp**
   3.1. С помощью манипулятора «мышь» выделить область в окне и записать изображение выделенной области на диск в формате BMP.
   Для указания местоположения файла *.bmp использовать стандартное окно диалога Windows для сохранения файла (библиотека MFC).
   3.2. Отобразить сохраненный на диске файл *.bmp в текущем окне Windows (команда меню **Tests►Image**), ***предварительно (!)*** указав координаты левого верхнего угла изображения с помощью ЛКМ (левая клавиша мыши).
   Для указания местоположения файла *.bmp использовать стандартное окно диалога Windows для открытия файла (библиотека MFC).

Растровый формат используется для хранения растровых данных. Файлы такого типа особенно хорошо подходят для хранения изображений реального мира, например, оцифрованных фотографий. Растровые файлы содержат битовую карту изображения и се спецификацию. Наиболее распространенные растровые форматы: BMP, TIFF, GIF, PCX, JPEG. Каждый из этих форматов имеет свои преимущества и свои недостатки.

Общий алгоритм работы загрузки изображения:

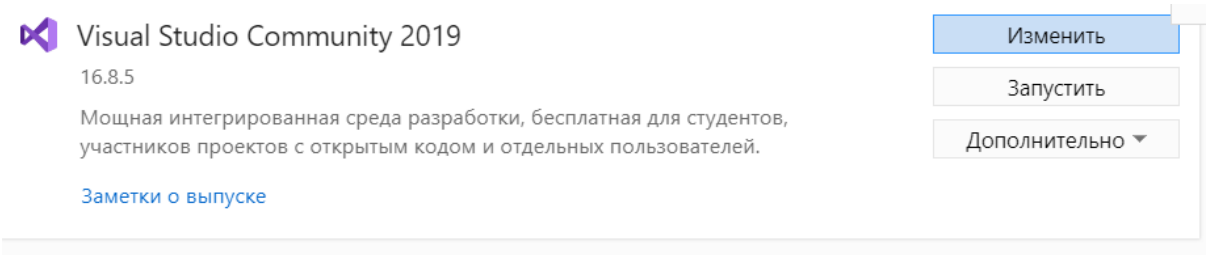
1. Получение параметров переданного изображения.
2. Выделение памяти для отображения переданного изображения.
3. Передача изображение в окно.

Общий алгоритм работы записи выделенной области изображения:

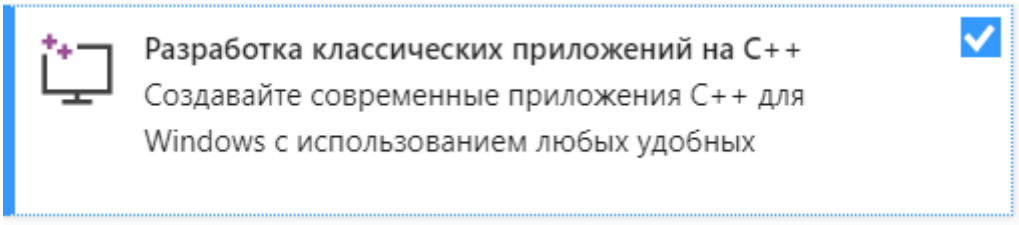
1. Получение координат точек левой верхней и правой нижней для захвата области выделения (при нажатии левой кнопки мыши мы захватываем точку левой вехней области изображения, при отпускании мы получаем правую нижнюю точку области изображения).
2. Получение информации хранимой в пределах данных двух точек.
3. Формирование файла для сохранения.
4. Сохранения файла в директорию.

Для выполнения этой лабораторной работой, вам требуется установить библиотеку MFC.

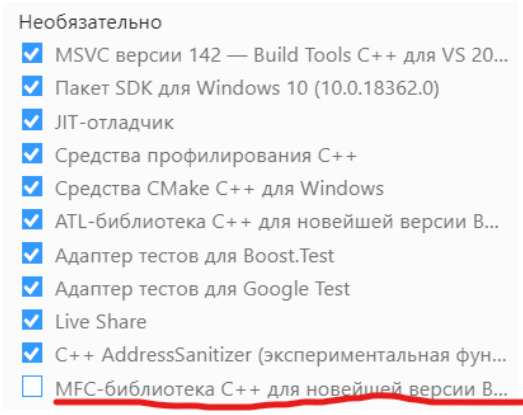
Заходим в Visual Studio Installer. Для добавления новых компонентов нажмите кнопку “Изменить”.



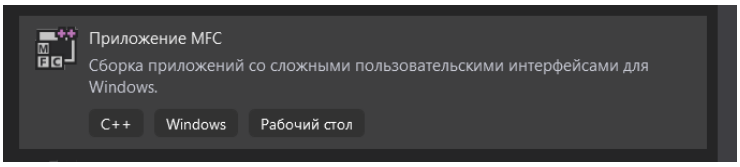
Выбираем пункт Разработка классических приложений на C++.



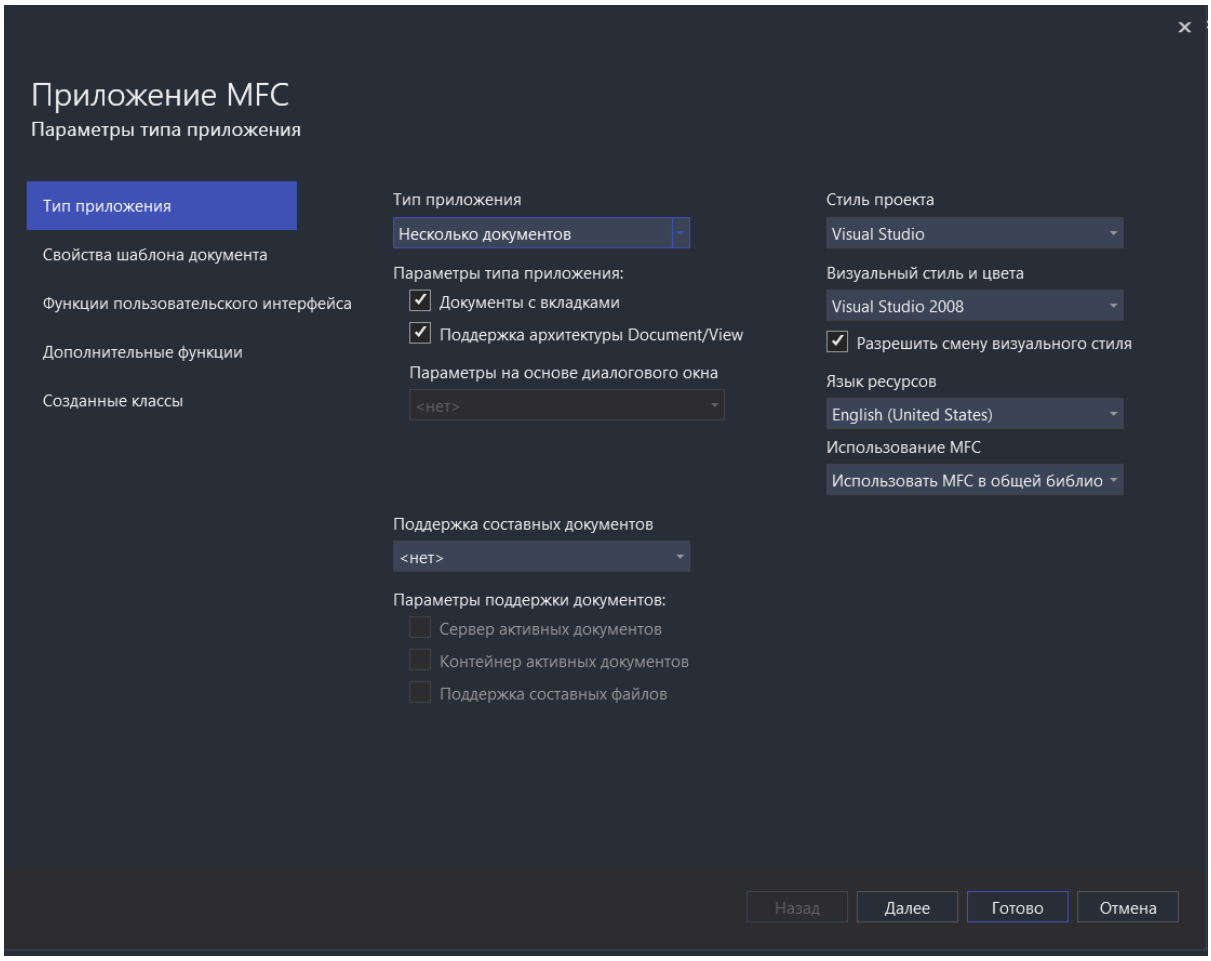
Далее в меню пакетов выбираем MFC-библиотеку C++.

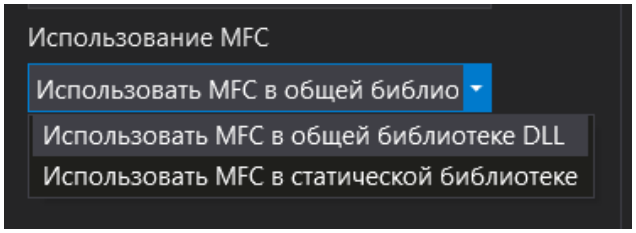


Создаём приложение MFC



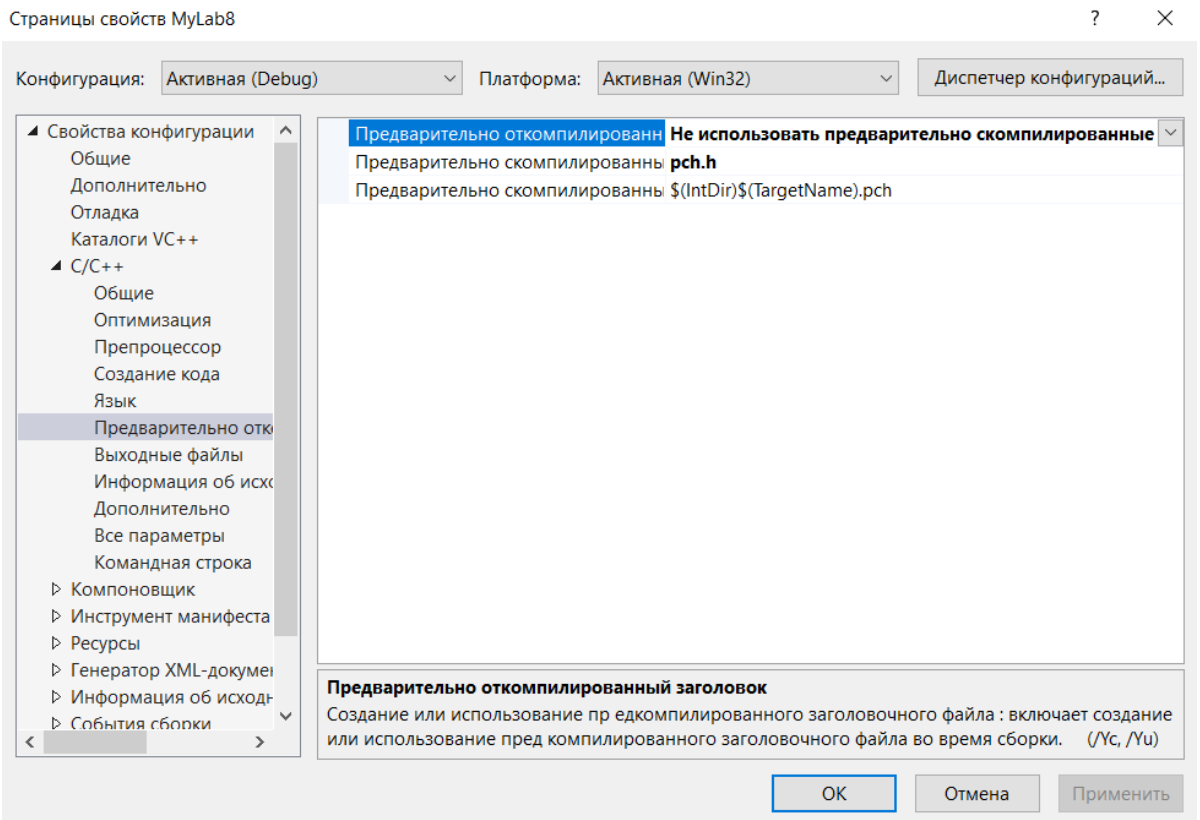
В настройках убеждаемся, что библиотека MFC будет использоваться в общей DLL





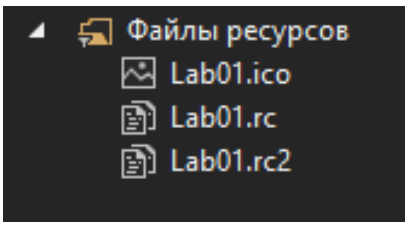
После удаления мусора, заходим в свойства проекта. Тыкаем на C/C++ и выбираем Предварительно откомпилированный или как там не вижу по скрину. И ставим что мы НЕ БУДЕМ использовать этот pch.h. Мы будем юзать stdafx для мерджина.

Теперь заходим в свойства проекта, C/C++, Предварительно откомпилированные и выбрать “Не использовать предварительно скомпилированные заголовки”.



Чтобы начать работу над проектом, требуется удалить все созданные, при создании приложения, файлы.

Вместе с этой лабораторной работой будет архив с нужными ресурсами. Их всего 3. Закидываем эти ресурсы в папку res, и добавляем в наш проект.
Получится вот так.



Их не надо менять, код, написанный далее. Он будет забинжен под эти ресурсы.

Далее создадим файл header.h и header.cpp, в котором будет служебная информация, а также подключения нужных нам заголовочных файлов.

header.h:

```
// stdafx.h: включите файл для добавления стандартных системных файлов
//или конкретных файлов проектов, часто используемых,
// но редко изменяемых

#pragma once

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN            // Исключите редко используемые компоненты из заголовков
Windows
#endif

#include "targetver.h"

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS      // некоторые конструкторы CString будут
явными

// отключает функцию скрытия некоторых общих и часто пропускаемых предупреждений MFC
#define _AFX_ALL_WARNINGS

#include <afxwin.h>         // основные и стандартные компоненты MFC
#include <afxext.h>         // расширения MFC

#include <FLOAT.H> // Для  DBL_MAX , DBL_MIN
#include <fstream>
#include <math.h>
#include "CMatrix.h"
#include <vector>

#include "LibGraph.h"



#ifndef _AFX_NO_OLE_SUPPORT
#include <afxdtctl.h>           // поддержка MFC для типовых элементов управления
Internet Explorer 4
```

```
#endif
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                    // поддержка MFC для типовых элементов управления
Windows
#endif // _AFX_NO_AFXCMN_SUPPORT

#include <afxcontrolbars.h>     // поддержка MFC для лент и панелей управления



#ifdef _UNICODE
#if defined _M_IX86
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='x86'
publicKeyToken='6595b64144ccf1df' language='*'\"")
#elif defined _M_X64
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='amd64'
publicKeyToken='6595b64144ccf1df' language='*'\"")
#else
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='*'
publicKeyToken='6595b64144ccf1df' language='*'\"")
#endif
#endif
```

Большинство заголовочных файлов будет подчеркнуто красным. Реализацию этих файлов напишем позже.

```
//{{NO_DEPENDENCIES}}
// Включаемый файл, созданный в Microsoft Visual C++.
// Используется Lab01.rc
//
#define IDD_ABOUTBOX                    100
#define IDP_OLE_INIT_FAILED             100
#define IDR_MAINFRAME                   128
#define IDR_Lab01TYPE                   130
#define IDB_BITMAP1                     310
#define ID_FILE_LOAD                    32771
#define ID_FILE_LOADINPOINT             32772
#define ID_FILE_SSAVE                   32773
#define ID_FILE_CLEAR                   32774
#define ID_32775                        32775

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        314
#define _APS_NEXT_COMMAND_VALUE         32776
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           310
#endif
#endif
```

targetver.h

```
#pragma once

// Включение SDKDDKVer.h обеспечивает определение самой последней доступной платформы Windows.

// Если требуется выполнить сборку приложения для предыдущей версии Windows, включите WinSDKVer.h и
// задайте для макроопределения _WIN32_WINNT значение поддерживаемой платформы перед вхождением SDKDDKVer.h.

#include <SDKDDKVer.h>
```

MainFrm.h

```
// MainFrm.h: интерфейс класса CMainFrame
//

#pragma once
#include "ChildView.h"

class CMainFrame : public CFrameWnd
{

public:
	CMainFrame() noexcept;
protected:
	DECLARE_DYNAMIC(CMainFrame)

// Атрибуты
public:

// Операции
public:

// Переопределение
public:
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
	virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO* pHandlerInfo);

// Реализация
public:
	virtual ~CMainFrame();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // встроенные члены панели элементов управления
	CStatusBar        m_wndStatusBar;
	CChildView    m_wndView;

// Созданные функции схемы сообщений
protected:
	afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
	afx_msg void OnSetFocus(CWnd *pOldWnd);
	DECLARE_MESSAGE_MAP()

};
```

MainFrm.cpp

```
// MainFrm.cpp: реализация класса CMainFrame
//

#include "header.h"
#include "Lab02.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
	ON_WM_CREATE()
	ON_WM_SETFOCUS()
END_MESSAGE_MAP()

static UINT indicators[] =
{
	ID_SEPARATOR,           // индикатор строки состояния
	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	ID_INDICATOR_SCRL,
};

// Создание или уничтожение CMainFrame

CMainFrame::CMainFrame() noexcept
{
	// TODO: добавьте код инициализации члена
}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;

	// создать представление для размещения рабочей области рамки
	if (!m_wndView.Create(nullptr, nullptr, AFX_WS_DEFAULT_VIEW, CRect(0, 0, 0, 0),
this, AFX_IDW_PANE_FIRST, nullptr))
	{
		TRACE0("Не удалось создать окно представлений\n");
		return -1;
	}

	if (!m_wndStatusBar.Create(this))
	{
		TRACE0("Не удалось создать строку состояния\n");
		return -1;      // не удалось создать
	}
```

```
	m_wndStatusBar.SetIndicators(indicators, sizeof(indicators)/sizeof(UINT));

	return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
	if( !CFrameWnd::PreCreateWindow(cs) )
		return FALSE;
	// TODO: изменить класс Window или стили посредством изменения
	//  CREATESTRUCT cs

	cs.style = WS_OVERLAPPED | WS_SYSMENU | WS_BORDER; // задаём вид окна

	cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
	cs.lpszClass = AfxRegisterWndClass(0);
	return TRUE;
}

// Диагностика CMainFrame

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CFrameWnd::Dump(dc);
}
#endif //_DEBUG


// Обработчики сообщений CMainFrame

void CMainFrame::OnSetFocus(CWnd* /*pOldWnd*/)
{
	// передача фокуса окну представления
	m_wndView.SetFocus();
}

BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO*
pHandlerInfo)
{
	// разрешить ошибки в представлении при выполнении команды
	if (m_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))
		return TRUE;
	// в противном случае выполняется обработка по умолчанию
	return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
}
```

TestBMP.h:

```
#pragma once
#include <Windows.h>
#define define_X 2
#define define_Y 2

int ShowBitMap(HWND hWnd, HANDLE hBit, int x, int y);
int ClientToBmp(HWND hwnd, RECT& r, char* name);
```

TestBMP.cpp:

```
#pragma once
#include "header.h"
#include "TestBMP.h"
#include <fstream>

int ShowBitMap(HWND hWnd, HANDLE hBit, int x, int y)
{
	// Функция отображает рисунок в заданной позиции окна
	// hWnd - дескриптор окна, куда выводится изображение
	// hBit - дескриптор рисунка
	// (x,y) - координаты левого верхнего угла изображения в окне вывода

	BITMAP BitMap;	// BITMAP - структура, которая определяет параметры растрового изображения
	GetObjectW(hBit, sizeof(BITMAP), &BitMap); // получаем параметры изображения в структуру BitMap
	int Height = BitMap.bmHeight;				// получаем высоту изображения
	int Width = BitMap.bmWidth;					// получаем ширину изображения
	HDC hdc = GetDC(hWnd);						// извлечение контекста изображения
	HDC hdcMem = CreateCompatibleDC(hdc);		// создание контекста памяти
	HBITMAP OldBitmap = (HBITMAP)SelectObject(hdcMem, hBit);	// в созданный контекст памяти заносим дескриптор битовой карты
	BitBlt(hdc, x, y, Width, Height, hdcMem, 0, 0, SRCCOPY);	// в окно, с которым связан контекст изображения, картинка переносится копированием
	SelectObject(hdcMem, OldBitmap);			// после копирования уничтожаются контексты памяти и изображения
	ReleaseDC(hWnd, hdc);
	return 0;
}

int ClientToBmp(HWND hWnd, RECT& r, char* name)
{
	// Сохранение рабочей области окна в файле name
	// hWnd - дескриптор окна, рабочая область которого сохраняется
	// r – область в  окне, которая сохраняется в файле
	// name - имя файла для сохранения

	// создаем файл
	HANDLE fh = CreateFile((LPCWSTR)name, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL | FILE_FLAG_SEQUENTIAL_SCAN, NULL);
	if (fh == INVALID_HANDLE_VALUE)				// если не создался
		return 2;

	BITMAPINFOHEADER bi;						// структура содержит описание изображения
	ZeroMemory(&bi, sizeof(BITMAPINFOHEADER));	// заполняет заголовок нулями
	bi.biSize = sizeof(BITMAPINFOHEADER);
	bi.biWidth = abs(r.right - r.left);
	bi.biHeight = abs(r.bottom - r.top);
	bi.biPlanes = 1;							// число плоскостей
	bi.biBitCount = 32;							// 32 глубина цветов (число битов на пиксель)
	// для вычисления размера изображения в байтах мы увеличиваем значение на строку до значения, кратного четырем.
	bi.biSizeImage = (bi.biWidth * bi.biBitCount + 31) / 32 * 4 * bi.biHeight;

	BITMAPFILEHEADER bmfHdr;					// заголовок, описывает тип
```

```
файла, размер, смещение области битов
	ZeroMemory(&bmfHdr, sizeof(BITMAPFILEHEADER));
	bmfHdr.bfType = 0x4D42;                                    //     BM          ('M'<<8)|'B';
	заполняем дисковый заголовок
	bmfHdr.bfSize = bi.biSizeImage + sizeof(BITMAPFILEHEADER) + bi.biSize;  //      размер
файла
	bmfHdr.bfReserved1 = bmfHdr.bfReserved2 = 0;
	bmfHdr.bfOffBits = (DWORD)sizeof(BITMAPFILEHEADER) + (DWORD)bi.biSize;  //    смещение
до начала пиксельных данных

	HDC hDC = GetDC(hWnd);                                     // контекст изображения
	HDC hDCMem = CreateCompatibleDC(hDC);           // получаем дескриптор памяти
	HBITMAP hBitmap = CreateCompatibleBitmap(hDC, bi.biWidth, bi.biHeight);       //
создаем битовую карту по размеру выделенного изображения
	HBITMAP oldBitmap = (HBITMAP)SelectObject(hDCMem, hBitmap);
	// в созданный контекст памяти вносит дескриптор битовой карты
	BitBlt(hDCMem,  0,  0,  bi.biWidth,  bi.biHeight,  hDC,  r.left,  r.top,  SRCCOPY);  //
копирует в память картинку
	hBitmap = (HBITMAP)SelectObject(hDCMem, oldBitmap);
	// восстанавливаем контекст памяти

	// выделяем место в памяти для того,чтобы функция GetDIBits() перенесла туда коды
цвета в DIB-формате.
	HANDLE hDIB = GlobalAlloc(GHND, bi.biSizeImage);
	char* lp = (char*)GlobalLock(hDIB);
	// из карты BitMap строки с нулевой по bi.biHeight функция пересылает в массив lp по
формату bi
	GetDIBits(hDC, hBitmap, 0, bi.biHeight, lp, (LPBITMAPINFO)&bi, DIB_RGB_COLORS);

	DWORD dwWritten = sizeof(BITMAPFILEHEADER);

	WriteFile(fh, (LPSTR)&bmfHdr, sizeof(BITMAPFILEHEADER), &dwWritten, NULL);
	// запись заголовка файла
	dwWritten = sizeof(BITMAPINFOHEADER);
	WriteFile(fh, (LPSTR)&bi, sizeof(BITMAPINFOHEADER), &dwWritten, NULL);
	// запись в файл загружаемого заголовка
	dwWritten = bi.biSizeImage;
	WriteFile(fh, lp, bi.biSizeImage, &dwWritten, NULL);
	// запись изображения на диск

	// освобождение памяти и закрытие файла
	GlobalUnlock(hDIB);
	GlobalFree(hDIB);

	DeleteObject(hBitmap);
	lp = NULL;
	CloseHandle(fh);

	ReleaseDC(hWnd, hDCMem);
	ReleaseDC(hWnd, hDC);

	DeleteDC(hDCMem);
	DeleteDC(hDC);
	if (dwWritten == 2) return 2;

	return 0;
}
```

ChildView.h:

```
// ChildView.h: интерфейс класса CChildView
//


#pragma once


// Окно CChildView

class CChildView : public CWnd
{
    // Создание
public:
    CChildView();

    // Атрибуты
public:
    CRect WinRect;
    CMatrix PView;
    CMatrix PLight;
    int Index;

    // Операции
public:

    // Переопределение
protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

    // Реализация
public:
    virtual ~CChildView();

    // Созданные функции схемы сообщений
protected:
    afx_msg void OnPaint();
    DECLARE_MESSAGE_MAP()
public:
    // действия при выборе пункта меню
    afx_msg void OnSphere_Mirror();
    afx_msg void OnSphere_Diffuse();
    afx_msg void OnSize(UINT nType, int cx, int cy);
};
```

ChildView.cpp:

```
// ChildView.cpp: реализация класса CChildView
//

#include "header.h"
#include "Lab02.h"
```

```
#include "ChildView.h"
#include "TestBMP.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CChildView

CChildView::CChildView()
{
}

CChildView::~CChildView()
{
}

// Реализация карты сообщений
BEGIN_MESSAGE_MAP(CChildView, CWnd)
	ON_WM_PAINT()
	ON_WM_LBUTTONDOWN()
		// сообщение ЛКМ нажата
	ON_WM_LBUTTONUP()
	// сообщение ЛКМ отпущена
	// сообщения меню выбора
	ON_COMMAND(ID_FILE_LOAD, &CChildView::OnFileLoad)                              //
загрузка файла
	ON_COMMAND(ID_FILE_LOADINPOINT, &CChildView::OnFileLoadInPoint)         // загрузка
файла в указанное место
	ON_COMMAND(ID_FILE_SSAVE, &CChildView::OnFileSave)                             //
сохранение области
	ON_COMMAND(ID_FILE_CLEAR, &CChildView::OnFileClear)                            //
очистка окна

END_MESSAGE_MAP()



// Обработчики сообщений CChildView

BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
	if (!CWnd::PreCreateWindow(cs))
		return FALSE;

	cs.dwExStyle |= WS_EX_CLIENTEDGE;
	cs.style &= ~WS_BORDER;
	cs.lpszClass = AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
		::LoadCursor(nullptr, IDC_ARROW), reinterpret_cast<HBRUSH>(COLOR_WINDOW+1),
nullptr);

	return TRUE;
}

void CChildView::OnPaint()
{
	CPaintDC dc(this); // контекст устройства для рисования
```

```
	if (theApp.index == 1)                    // если изображение загружено, происходит
отображение изображения
		ShowBitMap(theApp.hwnd, theApp.hBit, theApp.From.x, theApp.From.y);
	// вывод изображения
}

void CChildView::OnLButtonDown(UINT flags, CPoint point)
	// фиксация точки при нажатии ЛКМ
{
	theApp.From = point;
}




void CChildView::OnLButtonUp(UINT flags, CPoint point)
	// фиксация точки при отпускании ЛКМ
{
	theApp.To = point;
}


void CChildView::OnFileLoad()
		// загрузка файла по умолчанию
{
	theApp.index = 1;
	theApp.From.x = define_X;     // координаты левого верхнего угла изображения в окне
вывода
	theApp.From.y = define_Y;
	theApp.LoadImageBMP(theApp.From);
}


void CChildView::OnFileLoadInPoint()
		// загрузка файла в указанное место
{
	theApp.index = 1;
	theApp.LoadImageBMP(theApp.From);
}


void CChildView::OnFileSave()
		// запись указанной области изображения в файл
{
	theApp.SaveArea();
}

void CChildView::OnFileClear()
			//очистка области
{
	theApp.index = 0;
	Invalidate();
}
```

И реализуем файл запуска. Название файла должно соответствовать названию проекта.

MyLab2.h:

```
// Lab02.h: основной файл заголовка для приложения Lab01
//
#pragma once

#ifndef __AFXWIN_H__
#error "включить stdafx.h до включения этого файла в PCH"
#endif

#include "resource.h"       // основные символы


// CLab01App:
// Сведения о реализации этого класса: Lab01.cpp
//

class CLab01App : public CWinApp
{
public:
	CLab01App() noexcept;

	int index;		// "флаг" для проверки загрузки изображения
	HBITMAP hBit;	// дескриптор рисунка
	HWND hwnd;		// дескриптор окна, куда выводится изображение
	CPoint From;	// координаты курсора
	CPoint To;

// Переопределение
public:
	virtual BOOL InitInstance();
	virtual int ExitInstance();

// Реализация

public:
	afx_msg void OnAppAbout();

	void LoadImageBMP(CPoint point);    // загрузка изображения
	void SaveArea();                        // сохранить выделенную область окна
	DECLARE_MESSAGE_MAP()
};

extern CLab01App theApp;
```

MyLab2.cpp:

```
// Lab02.cpp: определяет поведение классов для приложения.
//
```

```
#include "header.h"
#include "afxwinappex.h"
#include "afxdialogex.h"
#include "Lab02.h"
#include "MainFrm.h"
#include "TestBMP.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CLab01App

BEGIN_MESSAGE_MAP(CLab01App, CWinApp)
END_MESSAGE_MAP()

// Clab2App construction

void CLab01App::LoadImageBMP(CPoint point)
	// загрузка изображения
{
	CFileDialog fileDialog((BOOL)TRUE);                                    //объект
класса выбора файла
	int result = fileDialog.DoModal();
	//запустить диалоговое окно
	if (result == IDOK)
	//если файл выбран
	{
		AfxMessageBox(fileDialog.GetPathName());                    // показать
полный путь
		CWnd *cwnd = AfxGetMainWnd();                                       //
получить указатель на активное главное окно приложения
		hwnd = cwnd->GetSafeHwnd();                                        //
Возвращает дескриптор окна m_hWnd или NULL если указатель thisNULL.
		hBit = (HBITMAP)LoadImage(NULL, fileDialog.GetPathName().GetBuffer(),
IMAGE_BITMAP,0, 0, LR_LOADFROMFILE);            //загрузка изображения
		ShowBitMap(hwnd, hBit, point.x, point.y);                   //вывод
изображения
	}
}

void CLab01App::SaveArea()
		// сохранение области изображения
{
	CFileDialog fileDialog((BOOL)FALSE, NULL, L"area.bmp");           //объект класса
выбора файла
	int result = fileDialog.DoModal();
	//запустить диалоговое окно
	if (result == IDOK)
	//если файл выбран
	{
		CWnd *cwnd = AfxGetMainWnd();
		HWND hwnd = cwnd->GetSafeHwnd();
		RECT r;
		//r - область в окне, куда выводится изображение
		if (From.x < To.x) {
			r.left = From.x;
			r.right = To.x;
```

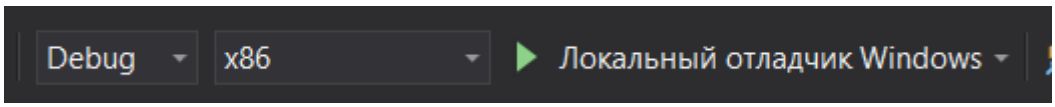
```
                  r.top = From.y;
                  r.bottom = To.y;
            }
            else {
                  r.left = To.x;
                  r.right = From.x;
                  r.top = To.y;
                  r.bottom = From.y;
            }
            ClientToBmp(hwnd, r, (char*)fileDialog.GetPathName().GetBuffer());      //
сохранение области изображения

      }
}
// Создание CLab01App

CLab01App::CLab01App() noexcept
{
      index = 0;

      SetAppID(_T("Lab01.AppID.NoVersion"));
}

// Единственный объект CLab01App

CLab01App theApp;


// Инициализация CLab01App

BOOL CLab01App::InitInstance()
{
      // InitCommonControlsEx() требуются для Windows XP, если манифест
      // приложения использует ComCtl32.dll версии 6 или более поздней версии для
включения
      // стилей отображения.  В противном случае будет возникать сбой при создании
любого окна.
      INITCOMMONCONTROLSEX InitCtrls;
      InitCtrls.dwSize = sizeof(InitCtrls);
      // Выберите этот параметр для включения всех общих классов управления, которые
необходимо использовать
      // в вашем приложении.
      InitCtrls.dwICC = ICC_WIN95_CLASSES;
      InitCommonControlsEx(&InitCtrls);

      CWinApp::InitInstance();

      // Инициализация библиотек OLE
      if (!AfxOleInit())
      {
            AfxMessageBox(IDP_OLE_INIT_FAILED);
            return FALSE;
      }

      AfxEnableControlContainer();

      EnableTaskbarInteraction(FALSE);

      // Для использования элемента управления RichEdit требуется метод
AfxInitRichEdit2()
```

```
	// AfxInitRichEdit2();

	// Стандартная инициализация
	// Если эти возможности не используются и необходимо уменьшить размер
	// конечного исполняемого файла, необходимо удалить из следующего
	// конкретные процедуры инициализации, которые не требуются
	// Измените раздел реестра, в котором хранятся параметры
	// TODO: следует изменить эту строку на что-нибудь подходящее,
	// например на название организации
	SetRegistryKey(_T("Локальные приложения, созданные с помощью мастера
приложений"));


	// Чтобы создать главное окно, этот код создает новый объект окна
	// рамки, а затем задает его как объект основного окна приложения
	CFrameWnd* pFrame = new CMainFrame;
	if (!pFrame)
		return FALSE;
	m_pMainWnd = pFrame;
	// создайте и загрузите рамку с его ресурсами
	pFrame->LoadFrame(IDR_MAINFRAME,
		WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, nullptr,
		nullptr);


	// Разрешить использование расширенных символов в горячих клавишах меню
	CMFCToolBar::m_bExtCharTranslation = TRUE;

	// Одно и только одно окно было инициализировано, поэтому отобразите и обновите
его
	pFrame->ShowWindow(SW_SHOW);
	pFrame->UpdateWindow();
	return TRUE;
}

int CLab01App::ExitInstance()
{
	//TODO: обработайте дополнительные ресурсы, которые могли быть добавлены
	AfxOleTerm(FALSE);

	return CWinApp::ExitInstance();
}

// Обработчики сообщений CLab01App


// Диалоговое окно CAboutDlg используется для описания сведений о приложении

class CAboutDlg : public CDialogEx
{
public:
	CAboutDlg() noexcept;

// Данные диалогового окна
#ifdef AFX_DESIGN_TIME
	enum { IDD = IDD_ABOUTBOX };
#endif

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // поддержка DDX/DDV
```

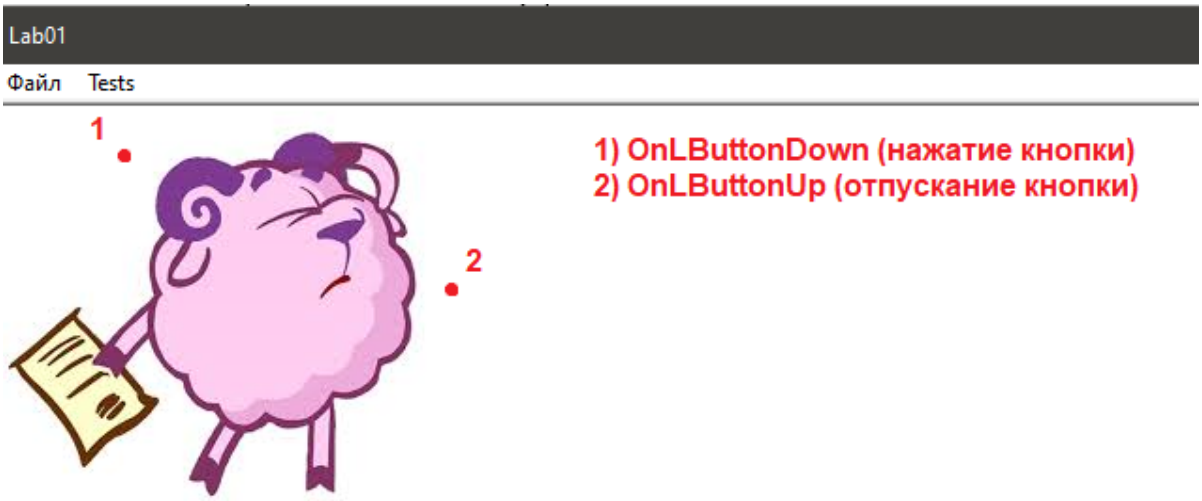
```
// Реализация
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() noexcept : CDialogEx(IDD_ABOUTBOX)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()

// Команда приложения для запуска диалога
void CLab01App::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}// Обработчики сообщений CLab01App
```

Теперь наше приложение готово к сборке.

Запускаем отладчик:



Получаем такой вот результат:



После сохранения выделенной области: